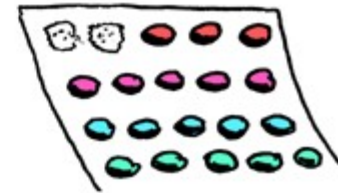


TURING MACHINES

Chapter 12

WHEN IT CAME TO EATING STRIPS OF CANDY BUTTONS, THERE WERE TWO MAIN STRATEGIES. SOME KIDS CAREFULLY REMOVED EACH BEAD, CHECKING CLOSELY FOR PAPER RESIDUE BEFORE EATING.



OTHERS TORE THE CANDY OFF HAPHAZARDLY, SWALLOWING LARGE SCRAPS OF PAPER AS THEY ATE.

THEN THERE WERE THE LONELY FEW OF US WHO MOVED BACK AND FORTH ON THE STRIP, EATING ROWS OF BEADS HERE AND THERE, PRETENDING WE WERE TURING MACHINES.



INTRODUCTION

- So, far we have looked at how computers can be used to solve problems.
- However, there are limits to computability.
- To study what computing agents can and cannot do, we must strip them down to bare essentials.
- Create an “ideal” computing agent, a model of a computing agent.

WHAT IS A MODEL?

- A model of something
 - Captures the essence of the real thing
 - Differs in scale from the real thing
 - Omits some details of the real thing
 - Lacks some of the functionality of the real thing
- Models can be used to predict behavior
 - Look at the model's behavior
 - May be safer, less expensive, less difficult than the real thing
- Models let us test something without building it.
- The model is only as good as the assumptions made in building it.

A MODEL OF A COMPUTING AGENT

What are the key features of a computing agent?

1. Read input data.
2. Store and retrieve information from memory.
3. Act in accordance with the given instructions.
4. Output results.

A MODEL OF A COMPUTING AGENT

Turing machine: a model of a computing agent proposed by Alan Turing in 1936.

1. The machine has a tape, infinite in both directions
 - Each cell of the tape holds one symbol
 - Tape alphabet: finite set of symbols for tape
 - Tape holds input and output

.	.	.	b	b	0	1	1	b	b	.	.	.
---	---	---	---	---	---	---	---	---	---	---	---	---

A Turing machine tape

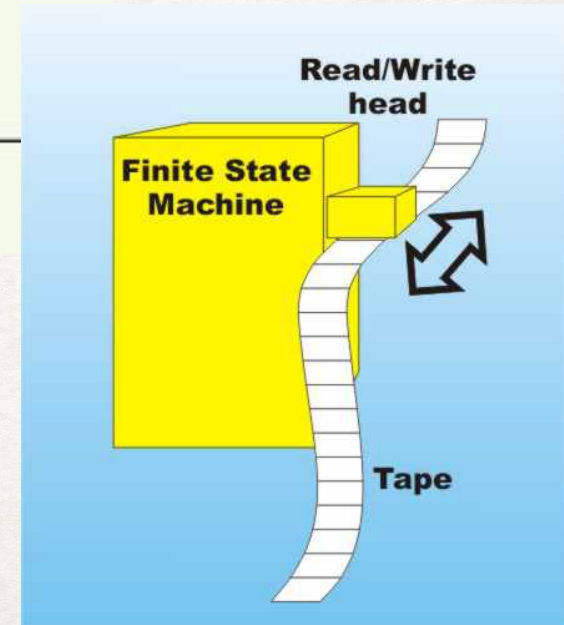
A MODEL OF A COMPUTING AGENT

2. The machine has a finite number of internal states, 1 to k



1 (current state of the machine)

A Turing machine configuration

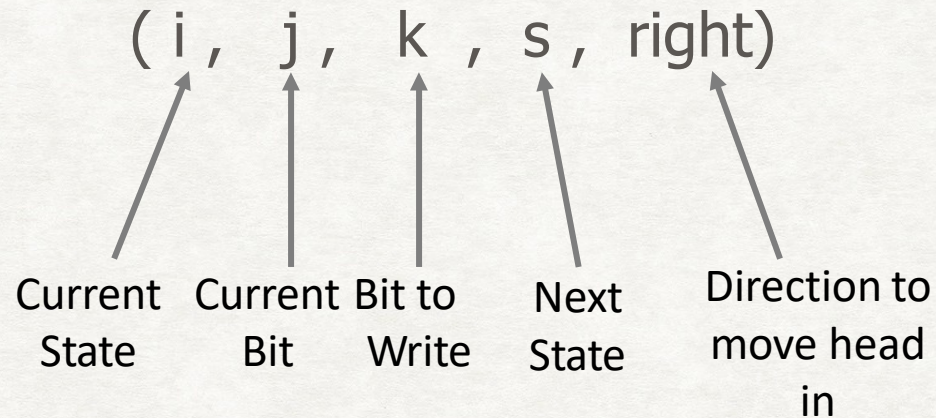


A MODEL OF A COMPUTING AGENT

3. The machine has instructions.

- Examine current tape symbol and current state
- Write new tape symbol, change state, and move left or right one cell

Example:



if (in state i) and (reading bit j) then

- Write bit k onto tape
- Change to state s
- Move right

A MODEL OF A COMPUTING AGENT

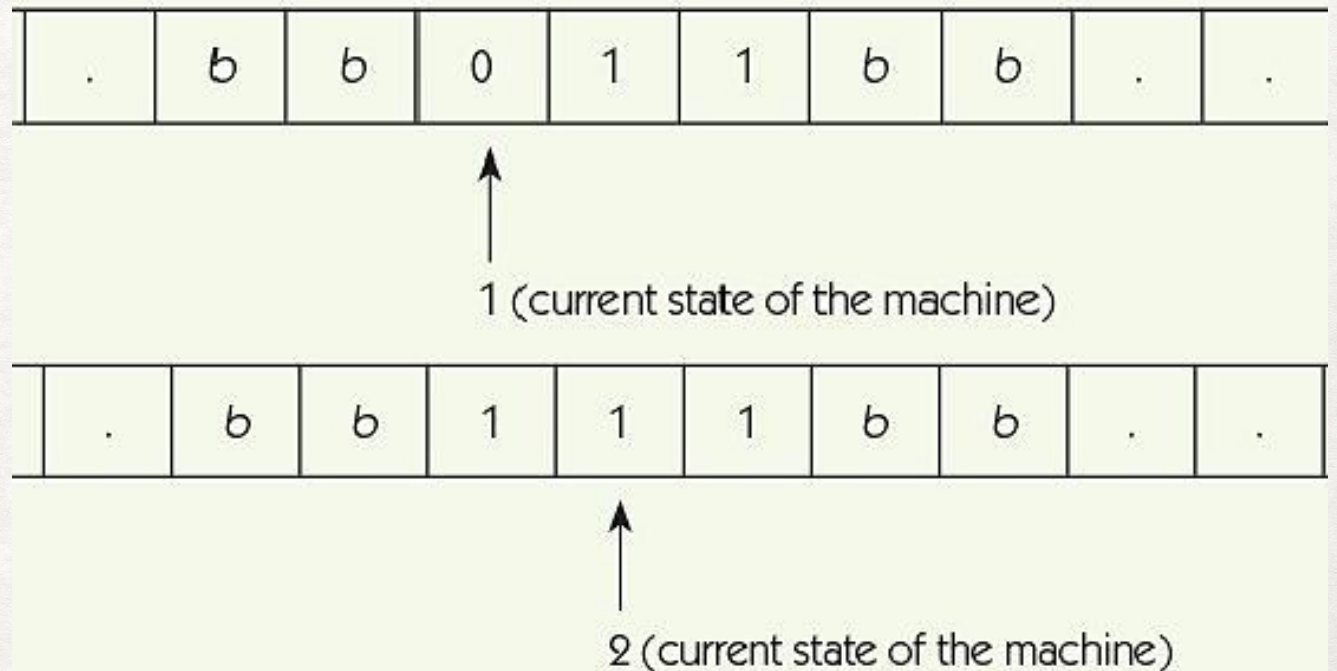
3. The machine has instructions.

- Examine current tape symbol and current state
- Write new tape symbol, change state, and move left or right one cell

Example:


(1 , 0 , 1 , 2, right)

if (state=1) and (input=0)
 write 1
 change into state 2
 move right



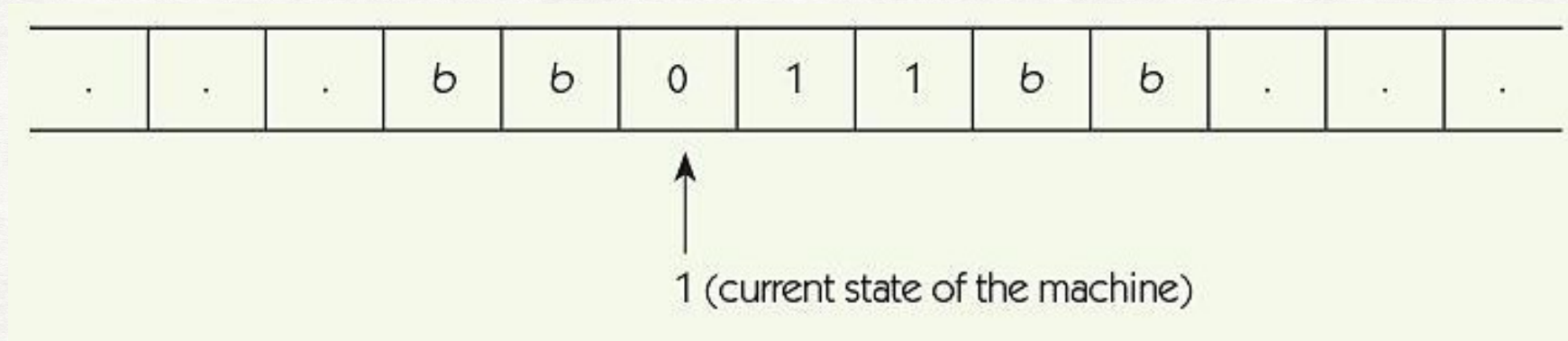
A MODEL OF A COMPUTING AGENT

Turing machine program: a set of rules

1. Must begin in state 1.
2. Machine always starts reading leftmost nonblank symbol in the input (unless specified otherwise).
3. Avoid ambiguity.
No two rules with the same state and input
 $(i, j, -, -, -)$
 $(i, j, -, -, -)$ 
4. If no rule applies, the machine halts.

EXERCISE

Suppose that the input to a Turing machine is as follows



And the set of instructions is as given below

(1,0,1,2,R)

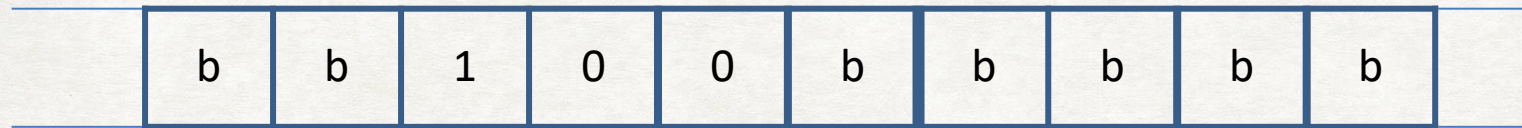
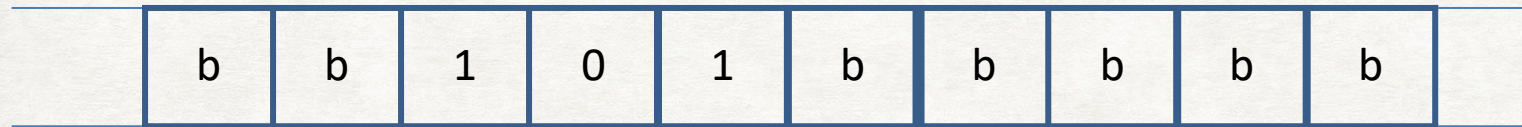
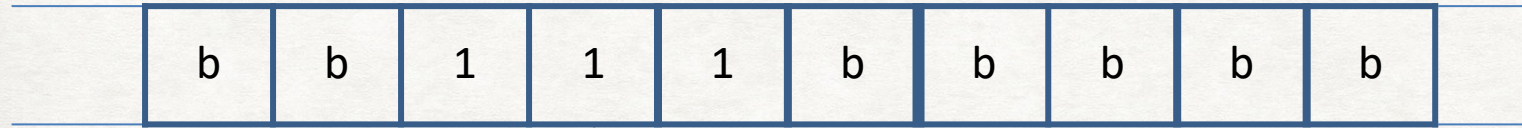
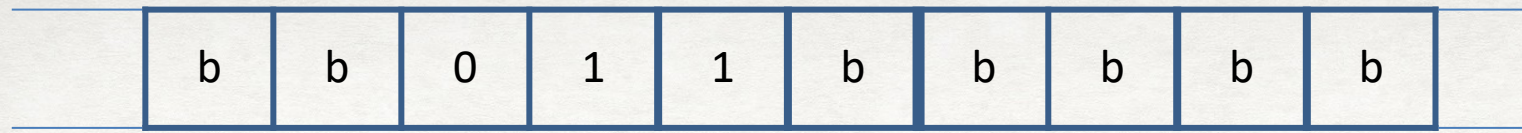
(1,1,1,2,R)

(2,0,1,2,R)

(2,1,0,2,R)

(2,b,b,3,L)

What is the output from the machine?



(1,0,1,2,R)

(1,1,1,2,R)

(2,0,1,2,R)

(2,1,0,2,R)

(2,b,b,3,L)

IS THE TURING MACHINE A GOOD MODEL OF A COMPUTING AGENT?

1. Read input data.
 - It reads input from its tape.
2. Store and retrieve information from memory.
 - It uses its tape as memory and can read from and write to it.
3. Act in accordance with the given instructions.
 - It takes actions based on its state and its input.
4. Output results.
 - It can produce output, on its tape.

Writing a set of Turing machine instructions is similar to writing a computer program in a programming language.

HOW MUCH OF AN APPROXIMATION OF A COMPUTING MODEL IS THE TURING MACHINE?

Like a general computing agent, it can follow many different sets of instructions and can do many different things.

- as we will shortly see.

It has eliminated the details of how the data is read or written to the tape, the meaning of the symbols on the tape and how the machine changes state.

In one respect though, it is larger than an actual real computer.

Can you guess what that is?

IS A TURING MACHINE PROGRAM A MODEL OF AN ALGORITHM?

An algorithm is defined as a *well-ordered collection of unambiguous and effectively computable operations* that, when executed, *produces a result and halts* in a finite amount of time.

1. Be a well-ordered collection.

- The starting state and position on the tape are well-defined.
- At most one rule can match a given state and position.
- It is well defined. What happens if no rule matches?

A Turing machine knows where to start, and what step to do next.

2. Consist of unambiguous and effectively computable operations.
3. Halt in a finite amount of time.
4. Produce a result.

IS A TURING MACHINE PROGRAM A MODEL OF AN ALGORITHM?

An algorithm (and hence its model) must:

1. Be a well-ordered collection.
2. Consist of unambiguous and effectively computable operations.
 - Turing machine instructions cannot be ambiguous.
 - Turing machine instructions completely specify what the machine must do.
3. Halt in a finite amount of time.
4. Produce a result.

IS A TURING MACHINE PROGRAM A MODEL OF AN ALGORITHM?

An algorithm (and hence its model) must:

1. Be a well-ordered collection.
2. Consist of unambiguous and effectively computable operations.
3. Halt in a finite amount of time.
 - Turing machines can go into infinite loops, like poor attempts at algorithms.
 - We can ensure halting for inputs within the scope of the problem.
 - “Universe of discourse” for the problem.
4. Produce a result.

IS A TURING MACHINE PROGRAM A MODEL OF AN ALGORITHM?

An algorithm (and hence its model) must:

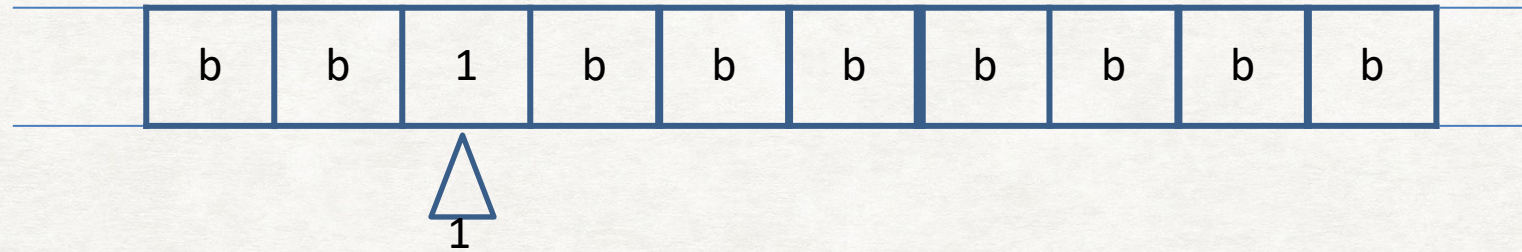
1. Be a well-ordered collection.
2. Consist of unambiguous and effectively computable operations.
3. Halt in a finite amount of time.
4. Produce a result.

The contents of the tape when the Turing machine halts is the output.

Turing machine halting in a finite amount of time.

For a Turing machine to halt,

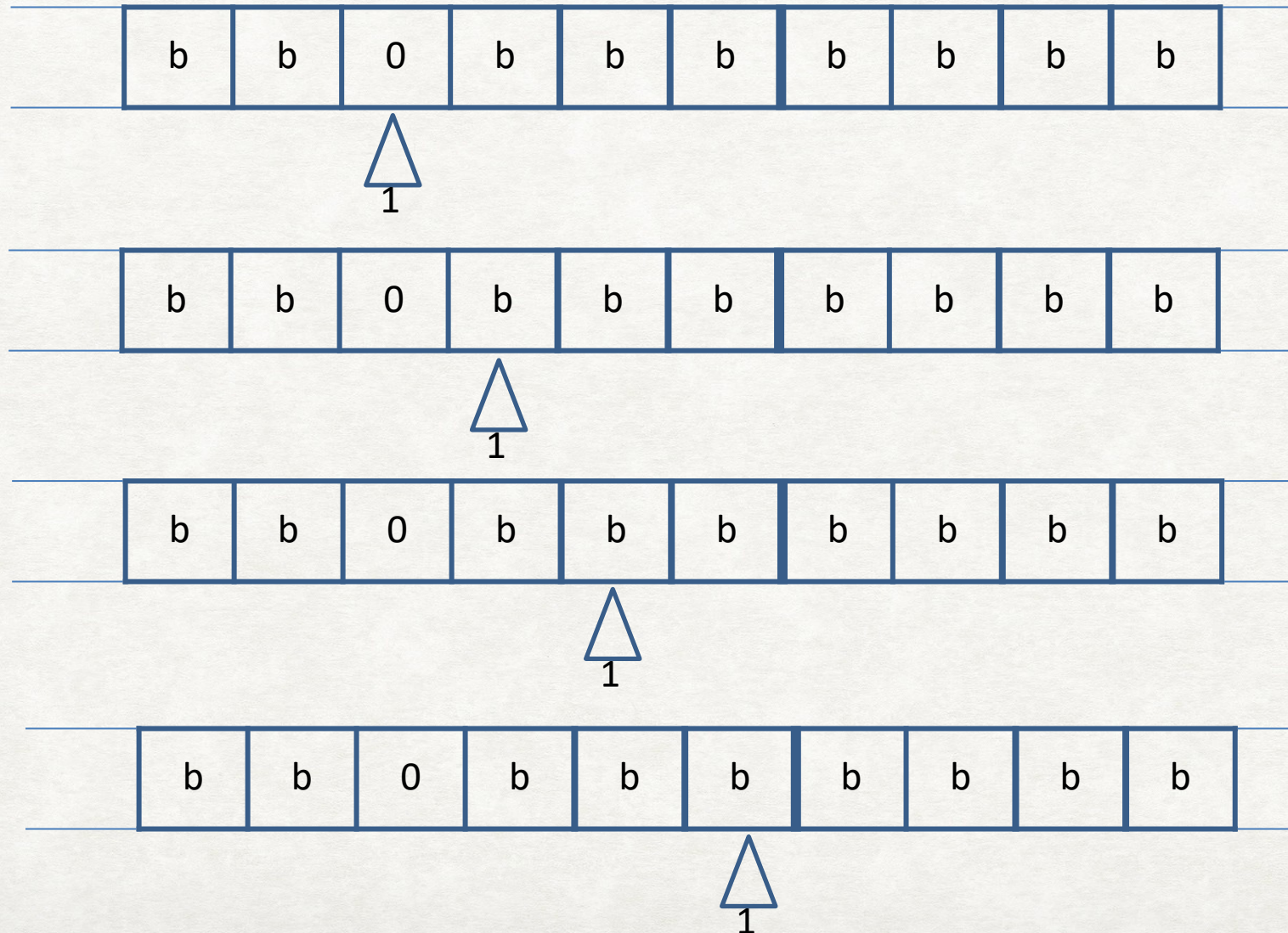
It must reach a configuration where no appropriate instruction exists.



$(1,0,0,1,R)$
 $(1,b,b,1,R)$

The machine halts immediately.

Now, assume we have the same set of instructions as before but with a single 0 as input.



$(1,0,0,1,R)$
 $(1,b,b,1,R)$

This machine will
never halt!

UNIVERSE OF DISCOURSE OF A PROBLEM

- The fact that a Turing machine may not halt on an input does not contradict with our definition of an algorithm!
- It simply confirms that there is always a “universe of discourse” connected with the problem we are trying to solve.
- For, example consider an algorithm for dividing a positive integer by another positive integer using repeated subtraction until the result is negative.

To compute $7/3$

$$7-3 = 4$$

$$4-3 = 1$$

$$1-3 < 0$$

To compute $7/(-3)$

$$7-(-3) = 10$$

$$10-(-3) = 13$$

$$13-(-3) = 16$$

$$16-(-3) = 19$$

A MODEL OF AN ALGORITHM

Thus, a Turing machine program is a good model of an algorithm!

In fact, there is no distinction made between a Turing machine as a computing agent and a Turing machine as a model of an algorithm (set of instructions for the machine).

We say we are going to “write a Turing machine” to solve a particular task, when what we actually mean is “we are going to write a Turing machine algorithm” to solve the task.

TURING MACHINE EXAMPLES

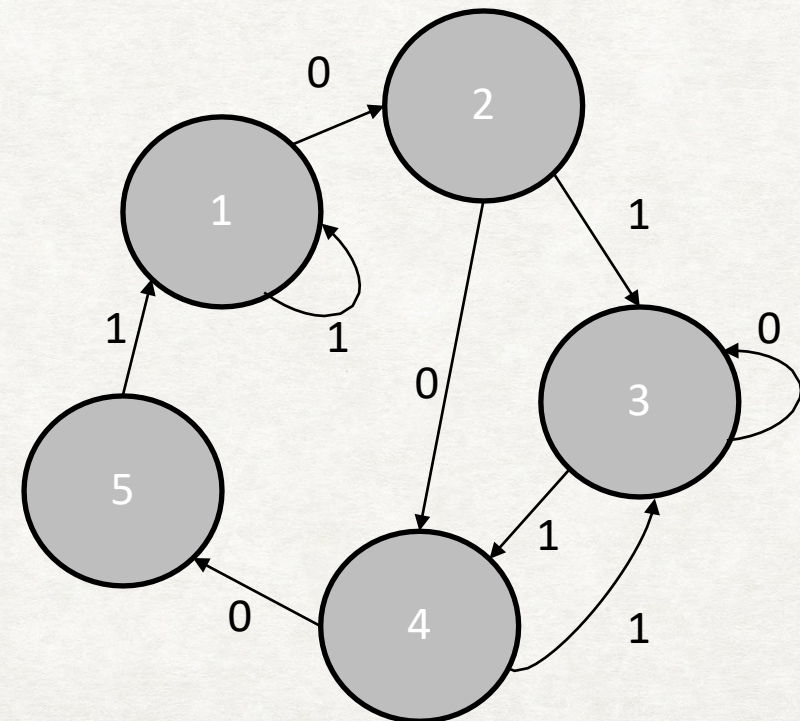
We use *pseudo codes* to write algorithms for computer programs.

We use a *state machine diagram* to write a Turing machine algorithm.

A state machine diagram consists of:

- A finite set of states

- Transitions between states, based on input



TURING MACHINE EXAMPLES

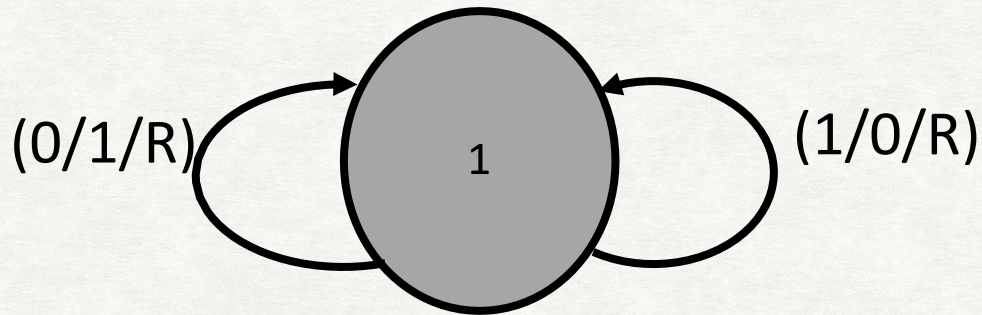
Examples showing what Turing machines can do

- Bit inverter
- Parity bit
- Unary increment
- Unary addition

BIT INVERTER

Given a string of 0s and 1s, change every 0 to a 1 and every 1 to a 0

Algorithm idea: move right, flipping each bit as it goes, and stop when you reach a blank

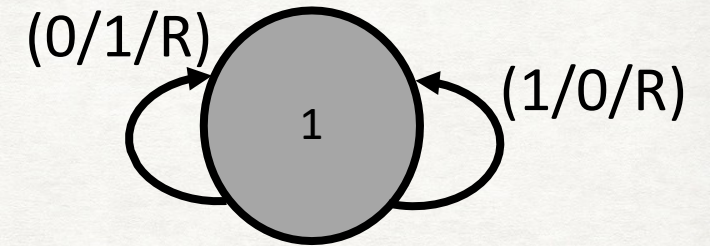


Rules:

(1, 0, 1, 1, R)

(1, 1, 0, 1, R)

BIT INVERTER MACHINE



Rules:

(1, 0, 1, 1, R)

(1, 1, 0, 1, R)

